Hoàng Như Vĩnh

VinhHN7@fe.edu.vn

Báo cáo Lab 3: Finite State Maching

Hoàng Như Vĩnh

Contents

[Phần 1: Máy trạng thái Moore và Mealy để xác định chuỗi bit ‘110’ 2](#_Toc165472101)

[Testbench cho hai thiết kế của máy trạng thái Moore và Mealy 2](#_Toc165472102)

[Phần 2: Thiết kế, mô hình hóa, và mô phỏng một máy trạng thái để xác định chuỗi 101. 6](#_Toc165472103)

[1. Thiết kế máy trạng thái Moore dùng để xác định chuỗi bit ‘101’. 6](#_Toc165472104)

[2. Mô hình hóa máy trạng thái trong mục 1 sử dụng ngôn ngữ VHDL 6](#_Toc165472105)

[3. Viết kịch bản kiểm tra (testbench) cho máy trạng thái Moore xác định chuỗi ‘101’ 8](#_Toc165472106)

[4. Mô phỏng thiết kế trong mục 2 với kịch bản kiểm tra trong mục 3. 12](#_Toc165472107)

[5. Thiết kế máy trạng thái Mealy dùng để tìm chuỗi “101” trong tín hiệu đầu vào. 12](#_Toc165472108)

# Phần 1: Máy trạng thái Moore và Mealy để xác định chuỗi bit ‘110’

## Testbench cho hai thiết kế của máy trạng thái Moore và Mealy

Testbench cho máy Moore và Mealy để xác định chuỗi bit ‘110’ được thể hiện như sau với hai pattern test cơ bản là ‘110’ và ‘1110’

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-------------------------------------------------------------------------------

ENTITY *fsm\_110\_detector\_tb* IS

END ENTITY fsm\_110\_detector\_tb;

-------------------------------------------------------------------------------

ARCHITECTURE *test* OF *fsm\_110\_detector\_tb* IS

  -- component ports

  SIGNAL clk                  : *STD\_LOGIC* := '1';

  SIGNAL rst\_n                : *STD\_LOGIC* := '0';

  SIGNAL data\_in              : *STD\_LOGIC* := '0';

  SIGNAL moore\_detect\_110\_out : *STD\_LOGIC*;

  SIGNAL mealy\_detect\_110\_out : *STD\_LOGIC*;

  CONSTANT PERIOD : *TIME* := 10 ns;

BEGIN -- ARCHITECTURE test

  -- component instantiation

  MOORE\_DUT : ENTITY work.moore\_110\_detector

    PORT MAP

    (

*clk*            => clk,

*rst\_n*          => rst\_n,

*data\_in*        => data\_in,

*detect\_110\_out* => moore\_detect\_110\_out);

  MEALY\_DUT : ENTITY work.mealy\_110\_detector

    PORT MAP

    (

*clk*            => clk,

*rst\_n*          => rst\_n,

*data\_in*        => data\_in,

*detect\_110\_out* => mealy\_detect\_110\_out);

  -- clock & reset generation

  clk\_process : PROCESS

  BEGIN

    WHILE now < 200 ns LOOP

      clk <= '0';

      WAIT FOR PERIOD / 2;

      clk <= '1';

      WAIT FOR PERIOD / 2;

    END LOOP;

    WAIT;

  END PROCESS clk\_process;

  -- Waveform generation and initialization block

  WaveGen\_Proc : PROCESS

  BEGIN

    -- Wait for initial setup time

    WAIT FOR 2 \* PERIOD + PERIOD / 4;

    -- Reset signal asserted

    rst\_n <= '1';

    -- Wait for clock rising edge

    WAIT UNTIL rising\_edge(clk);

    -- Wait for quarter period

    WAIT FOR PERIOD / 4;

    -- Set initial data\_in value

    data\_in <= '1';

    -- Generate waveform pattern

    WAIT FOR PERIOD \* 2;

    data\_in <= '0';

    -- Assertion for Mealy state detect output after pattern generation

    WAIT FOR 2 ns;

    ASSERT mealy\_detect\_110\_out = '1' REPORT "Mealy: 110 detected failed" SEVERITY ERROR;

    -- Assertion for Moore state detect output after pattern generation

    WAIT UNTIL rising\_edge(clk);

    WAIT FOR 2 ns;

    ASSERT moore\_detect\_110\_out = '1' REPORT "Moore: 110 detected failed" SEVERITY ERROR;

    -- Repeat waveform generation and testing for a different pattern

    -- Generate second waveform pattern

    WAIT FOR 3 \* PERIOD;

    data\_in <= '1';

    WAIT FOR 3 \* PERIOD;

    data\_in <= '0';

    -- Assertion for Mealy state detect output after second pattern generation

    WAIT FOR 2 ns;

    ASSERT mealy\_detect\_110\_out = '1' REPORT "Mealy: 110 detected failed" SEVERITY ERROR;

    -- Assertion for Moore state detect output after second pattern generation

    WAIT UNTIL rising\_edge(clk);

    WAIT FOR 2 ns;

    ASSERT moore\_detect\_110\_out = '1' REPORT "Moore: 110 detected failed" SEVERITY ERROR;

    -- End of testbench report

    REPORT "END fsm\_110\_detector\_tb TESTBENCH";

    -- Wait indefinitely

    WAIT;

  END PROCESS WaveGen\_Proc;

END ARCHITECTURE test;

-------------------------------------------------------------------------------

CONFIGURATION fsm\_110\_detector\_tb\_test\_cfg OF fsm\_110\_detector\_tb IS

  FOR test

  END FOR;

END fsm\_110\_detector\_tb\_test\_cfg;

-------------------------------------------------------------------------------

Waveform của Testbench được trích xuất ra như sau

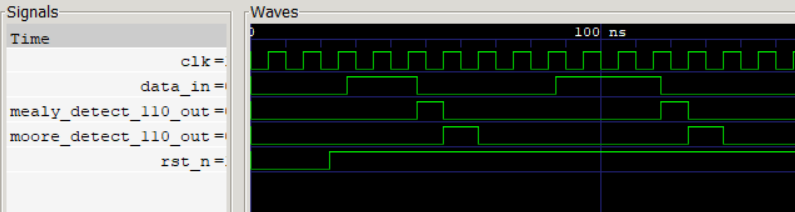


Figure 1 - Testbench cho máy Moore và Mealy xác định chuỗi bit '110'

Kết quả testbench cho thấy máy Moore và Mealy hoạt động đúng theo sơ đồ trạng thái.

Đối với máy Moore, moore\_detect\_110\_out sẽ chuyển trạng thái lên ‘1’ ở sườn lên của clk sau khi xác định được chuỗi ‘110’. Nói cách khác, tín hiệu moore\_detect\_110\_out sẽ hoạt động đồng bộ với clk. Khi thiết kế với máy Moore, thiết kế sẽ đòi hỏi nhiều state hơn, dẫn đến đòi hỏi nhiều phần cứng hơn.

Đối với máy Mealy, mealy\_detect\_110\_out sẽ chuyển trạng thái lên ‘1’ ngày sau khi data\_in thay đổi về ‘0’. Nói cách khác, tín hiệu mealy\_detect\_110\_out sẽ hoạt động bất đồng bộ với clk. Đối với máy Mealy sẽ yêu cầu ít phần cứng hơn, như vậy sẽ tiết kiệm được phần cứng trong lúc thiết kế.

# Phần 2: Thiết kế, mô hình hóa, và mô phỏng một máy trạng thái để xác định chuỗi 101.

## Thiết kế máy trạng thái Moore dùng để xác định chuỗi bit ‘101’.

Thiết kế của máy Moore dùng để xác định chuỗi bit ‘101’ như sau

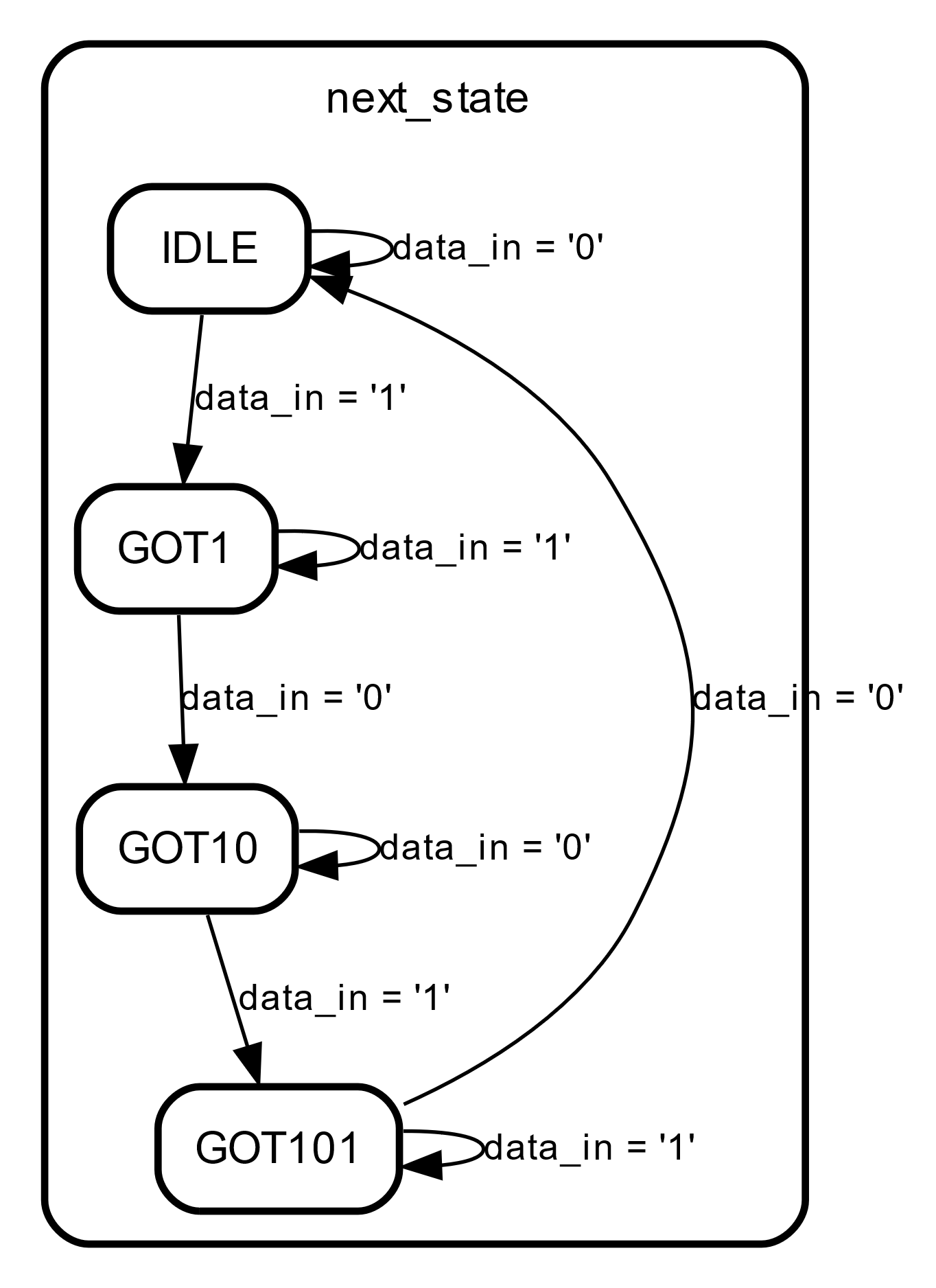


Figure 2 - Thiết kế máy Moore xác định chuỗi '101'

## Mô hình hóa máy trạng thái trong mục 1 sử dụng ngôn ngữ VHDL

Mã nguồn của moore\_101\_detector được viết như sau

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY *moore\_101\_detector* IS

  PORT (

    clk            : IN *STD\_LOGIC*;

    rst\_n          : IN *STD\_LOGIC*;

    data\_in        : IN *STD\_LOGIC*;

    detect\_101\_out : OUT *STD\_LOGIC*);

END moore\_101\_detector;

ARCHITECTURE *behavioral* OF *moore\_101\_detector* IS

  TYPE *state\_t* IS (*IDLE*, *GOT1*, *GOT10*, *GOT101*);

  SIGNAL current\_state : *state\_t*;

  SIGNAL next\_state    : *state\_t*;

BEGIN -- behavioral

  -- purpose: state register

  -- type   : sequential

  -- inputs : clk, rst\_n, next\_state

  -- outputs: current\_state

  REG : PROCESS (clk, rst\_n)

  BEGIN               -- PROCESS REG

    IF rst\_n = '0' THEN -- asynchronous reset (active low)

      current\_state <= IDLE;

    ELSIF clk'EVENT AND clk = '1' THEN -- rising clock edge

      current\_state <= next\_state;

    END IF;

  END PROCESS REG;

  -- purpose: calculate the transition of the FSM

  -- type   : combinational

  -- inputs : data\_in, current\_state

  -- outputs: next\_state

  NEXTSTATE : PROCESS (data\_in, current\_state)

  BEGIN -- PROCESS FSM

    CASE current\_state IS

      WHEN IDLE =>

        IF data\_in = '1' THEN

          next\_state <= GOT1;

        ELSIF data\_in = '0' THEN

          next\_state <= IDLE;

        END IF;

      WHEN GOT1 =>

        IF data\_in = '0' THEN

          next\_state <= GOT10;

        ELSIF data\_in = '1' THEN

          next\_state <= GOT1;

        END IF;

      WHEN GOT10 =>

        IF data\_in = '1' THEN

          next\_state <= GOT101;

        ELSIF data\_in = '0' THEN

          next\_state <= GOT10;

        END IF;

      WHEN GOT101 =>

        IF data\_in = '1' THEN

          next\_state <= GOT101;

        ELSIF data\_in = '0' THEN

          next\_state <= IDLE;

        END IF;

      WHEN OTHERS => NULL;

    END CASE;

  END PROCESS NEXTSTATE;

  detect\_101\_out <= '1' WHEN current\_state = GOT101 ELSE

    '0';

END behavioral;

## Viết kịch bản kiểm tra (testbench) cho máy trạng thái Moore xác định chuỗi ‘101’

Testbench cho máy Moore xác định chuỗi ‘101’ được viết như sau với data\_in = ‘101’ và ‘101101’

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-------------------------------------------------------------------------------

ENTITY *fsm\_101\_detector\_tb* IS

END ENTITY fsm\_101\_detector\_tb;

-------------------------------------------------------------------------------

ARCHITECTURE *test* OF *fsm\_101\_detector\_tb* IS

  -- component ports

  SIGNAL clk                  : *STD\_LOGIC* := '1';

  SIGNAL rst\_n                : *STD\_LOGIC* := '0';

  SIGNAL data\_in              : *STD\_LOGIC* := '0';

  SIGNAL moore\_detect\_101\_out : *STD\_LOGIC*;

  SIGNAL mealy\_detect\_101\_out : *STD\_LOGIC*;

  CONSTANT PERIOD             : *TIME* := 10 ns;

BEGIN -- ARCHITECTURE test

  -- component instantiation

  MOORE\_DUT : ENTITY work.moore\_101\_detector

    PORT MAP

    (

*clk*            => clk,

*rst\_n*          => rst\_n,

*data\_in*        => data\_in,

*detect\_101\_out* => moore\_detect\_101\_out);

  MEALY\_DUT : ENTITY work.mealy\_101\_detector

    PORT MAP

    (

*clk*            => clk,

*rst\_n*          => rst\_n,

*data\_in*        => data\_in,

*detect\_101\_out* => mealy\_detect\_101\_out

    );

  -- clock & reset generation

  clk\_process : PROCESS

  BEGIN

    WHILE now < 200 ns LOOP

      clk <= '0';

      WAIT FOR PERIOD / 2;

      clk <= '1';

      WAIT FOR PERIOD / 2;

    END LOOP;

    WAIT;

  END PROCESS clk\_process;

  -- Waveform generation and initialization block

  WaveGen\_Proc : PROCESS

  BEGIN

    -- Wait for initial setup time

    WAIT FOR 2 \* PERIOD + PERIOD / 4;

    -- Reset signal asserted

    rst\_n <= '1';

    -- Wait for clock rising edge

    WAIT UNTIL rising\_edge(clk);

    -- Wait for quarter period

    WAIT FOR PERIOD / 4;

    -- Generate waveform pattern

    data\_in <= '1';

    WAIT FOR PERIOD \* 1;

    data\_in <= '0';

    WAIT FOR PERIOD \* 1;

    data\_in <= '1';

    -- Assertion for Moore state detect output after pattern generation

    WAIT UNTIL rising\_edge(clk);

    WAIT FOR 2 ns;

    ASSERT moore\_detect\_101\_out = '1' REPORT "Moore: 101 detected failed" SEVERITY ERROR;

    -- reset data\_in

    data\_in <= '0';

    WAIT FOR 5 \* PERIOD;

    -- Repeat waveform generation and testing for a pattern 10101

    data\_in <= '1';

    WAIT FOR PERIOD \* 1;

    data\_in <= '0';

    WAIT FOR PERIOD \* 1;

    data\_in <= '1';

    WAIT FOR PERIOD \* 1;

    -- WAIT UNTIL rising\_edge(clk);

    -- WAIT FOR 2 ns;

    -- ASSERT moore\_detect\_101\_out = '1' REPORT "Moore: 101 detected" SEVERITY ERROR;

    data\_in <= '0';

    WAIT FOR PERIOD \* 1;

    data\_in <= '1';

    WAIT UNTIL rising\_edge(clk);

    WAIT FOR 2 ns;

    ASSERT moore\_detect\_101\_out = '0' REPORT "Moore: 101 NOT detected" SEVERITY ERROR;

    -- End of testbench report

    REPORT "END fsm\_101\_detector\_tb TESTBENCH";

    -- Wait indefinitely

    WAIT;

  END PROCESS WaveGen\_Proc;

END ARCHITECTURE test;

-------------------------------------------------------------------------------

CONFIGURATION fsm\_101\_detector\_tb\_test\_cfg OF fsm\_101\_detector\_tb IS

  FOR test

  END FOR;

END fsm\_101\_detector\_tb\_test\_cfg;

-------------------------------------------------------------------------------

## Mô phỏng thiết kế trong mục 2 với kịch bản kiểm tra trong mục 3.

Waveform của Testbench cho máy Moore được thể hiện như sau

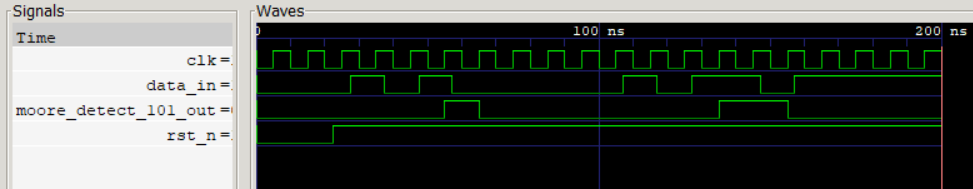


Figure 3 - Waveform cho máy Moore xác định chuỗi '101' với data\_in = ‘101’ và ‘101101’

Dựa theo Waveform, khi data\_in = ‘10101’, máy Moore xác định là 1 chuỗi ‘101’.

## Thiết kế máy trạng thái Mealy dùng để tìm chuỗi “101” trong tín hiệu đầu vào.

Thiết kế của máy trạng thái Mealy xác định chuỗi ‘101’ được thiết kế như sau

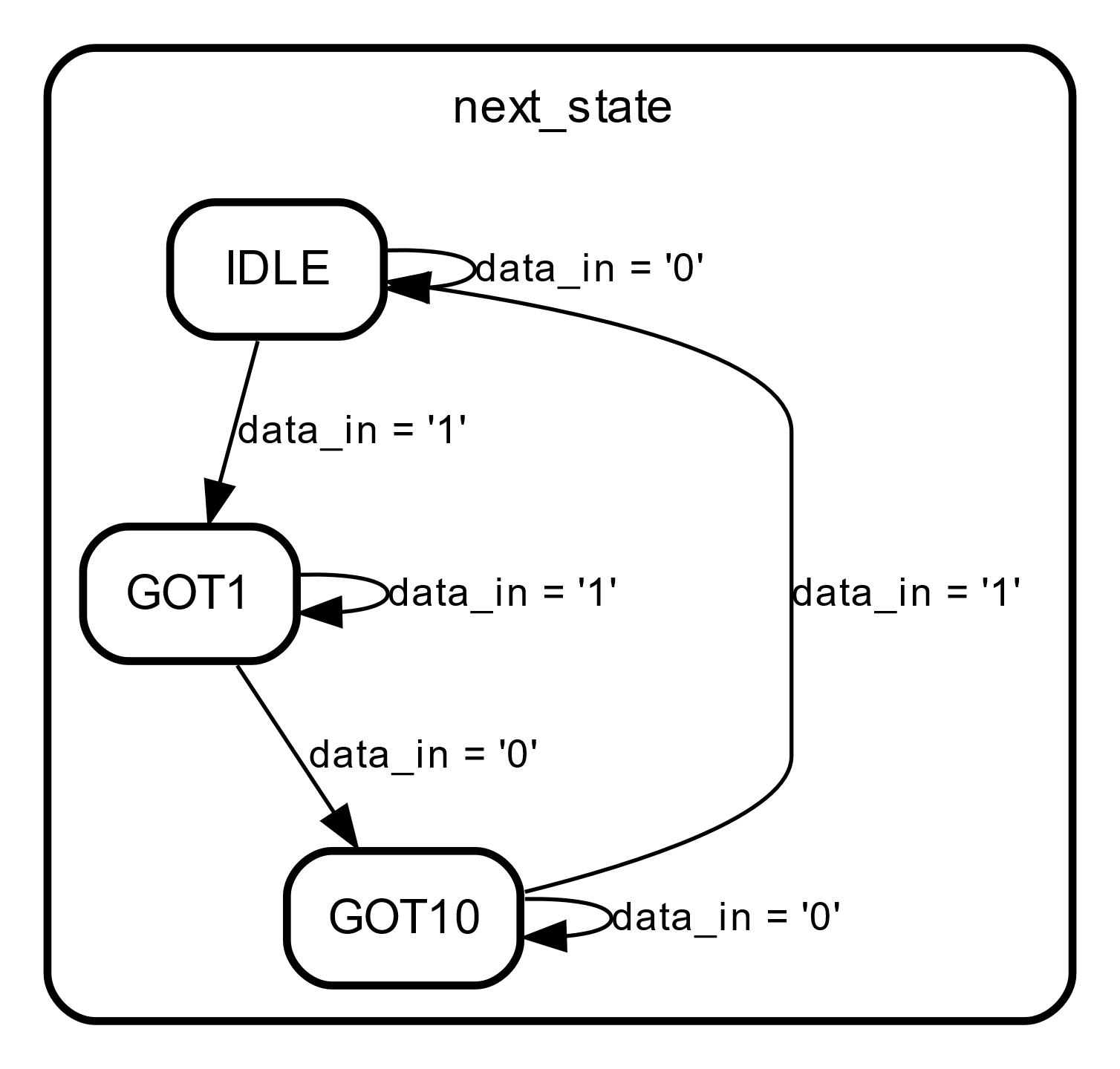


Figure 4 - Thiết kế máy Mealy xác định chuỗi '101'

Mã nguồn Máy trạng thái Mealy xác định chuỗi ‘101’ được viết như sau

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY *mealy\_101\_detector* IS

  PORT (

    clk            : IN *STD\_LOGIC*;

    rst\_n          : IN *STD\_LOGIC*;

    data\_in        : IN *STD\_LOGIC*;

    detect\_101\_out : OUT *STD\_LOGIC*);

END mealy\_101\_detector;

ARCHITECTURE *behavioral* OF *mealy\_101\_detector* IS

  TYPE *state* IS (*IDLE*, *GOT1*, *GOT10*);

  SIGNAL current\_state : *state*;

  SIGNAL next\_state    : *state*;

BEGIN -- behavioral

  -- purpose: state register

  -- type   : sequential

  -- inputs : clk, rst\_n, next\_state

  -- outputs: current\_state

  REG : PROCESS (clk, rst\_n)

  BEGIN               -- PROCESS REG

    IF rst\_n = '0' THEN -- asynchronous reset (active low)

      current\_state <= IDLE;

    ELSIF clk'EVENT AND clk = '1' THEN -- rising clock edge

      current\_state <= next\_state;

    END IF;

  END PROCESS REG;

  -- purpose: calculate the transition of the FSM

  -- type   : combinational

  -- inputs : data\_in, current\_state

  -- outputs: next\_state

  NEXTSTATE : PROCESS (data\_in, current\_state)

  BEGIN -- PROCESS FSM

    CASE current\_state IS

      WHEN IDLE =>

        IF data\_in = '1' THEN

          next\_state <= GOT1;

        ELSIF data\_in = '0' THEN

          next\_state <= IDLE;

        END IF;

      WHEN GOT1 =>

        IF data\_in = '0' THEN

          next\_state <= GOT10;

        ELSIF data\_in = '1' THEN

          next\_state <= GOT1;

        END IF;

      WHEN GOT10 =>

        IF data\_in = '0' THEN

          next\_state <= GOT10;

        ELSIF data\_in = '1' THEN

          next\_state <= IDLE;

        END IF;

      WHEN OTHERS => NULL;

    END CASE;

  END PROCESS NEXTSTATE;

  detect\_101\_out <= '1' WHEN (current\_state = GOT10 AND data\_in = '1') ELSE

    '0';

END behavioral;

Với thiết này với chỗi ‘10101’, máy Mealy xác định được 2 chuỗi ‘101’ thay vì 1 chuỗi đối với máy Moore.

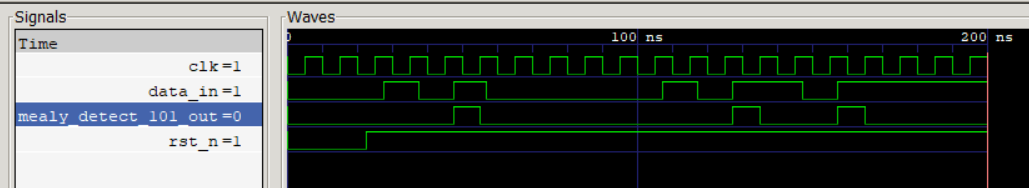


Figure 5 - Waveform của máy Mealy xác định chuỗi '101' với data\_in = ‘101’ và '101101'